

# NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



## THESIS

### A DESCRIPTION OF THE PANSAT COMMAND LANGUAGE

by

Troy M. Nichols

September, 1995

Thesis Advisor:

I. Michael Ross

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 1

19960305 104

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE A DESCRIPTION OF THE PANSAT COMMAND LANGUAGE		5. FUNDING NUMBERS		
6. AUTHOR(S) Nichols, Troy M.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (maximum 200 words)  This thesis details the PANSAT Command Language (PCL). The PCL is used for the efficient and reliable operation and commanding of PANSAT. This thesis describes each command of the PCL in detail and precisely defines the intended behavior and anticipated response a command will have on the various states of PANSAT. This thesis develops the user-syntax as well as the protocol-syntax for the PCL and begins to identify potential contingency scenarios. Numerous options and issues that were considered during the PCL development are provided and explanations given as to why certain options were chosen, while others were abandoned.				
14. SUBJECT TERMS PANSAT, Telemetry, Commanding, Satellite			15. NUMBER OF PAGES 156	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18 298-102



Approved for public release; distribution is unlimited.

**A DESCRIPTION OF THE PANSAT  
COMMAND LANGUAGE**

Troy M. Nichols  
Lieutenant, United States Navy  
B.S., Michigan State University, 1988

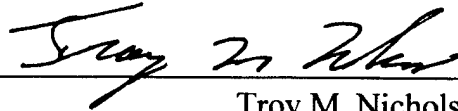
Submitted in partial fulfillment  
of the requirements for the degree of

**MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY  
(SPACE SYSTEMS OPERATIONS)**


from the


**NAVAL POSTGRADUATE SCHOOL  
September, 1995**

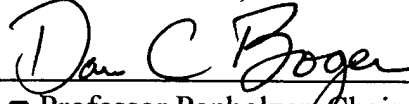
Author:

  
Troy M. Nichols

Approved by:

  
I. Michael Ross, Thesis Advisor

  
Daniel Sakoda, Second Reader

  
FOR Professor Panholzer, Chairman  
Space Systems Academic Group



## **ABSTRACT**

This thesis details the PANSAT Command Language (PCL). The PCL is used for the efficient and reliable operation and commanding of PANSAT. This thesis describes each command of the PCL in detail and precisely defines the intended behavior and anticipated response a command will have on the various states of PANSAT. This thesis develops the user-syntax as well as the protocol-syntax for the PCL and begins to identify potential contingency scenarios. Numerous options and issues that were considered during the PCL development are provided and explanations given as to why certain options were chosen, while others were abandoned.



## **ACKNOWLEDGMENT**

The author would like to thank Jim Horning of the Space Systems Academic Group, for his patience, time, and expert tutelage, without which this thesis would not have been possible.





## TABLE OF CONTENTS

I. INTRODUCTION .....	1
A. PANSAT MISSION DESCRIPTION .....	1
B. PANSAT COMMAND LANGUAGE .....	3
II. PANSAT HARDWARE ARCHITECTURE .....	5
A. RF SUBSYSTEM .....	6
B. DIGITAL CONTROL SUBSYSTEM (DCS) .....	8
C. THE PCB AND THE PCL .....	10
D. ELECTRICAL POWER SUBSYSTEM (EPS) .....	12
III. TELEMETRY .....	15
IV. GROUND STATION HARDWARE AND SOFTWARE ARCHITECTURE ....	17
A. TRANSCEIVER SUBSYSTEM .....	17
B. COMPUTER SUBSYSTEM .....	18
C. ANTENNA SUBSYSTEM .....	18
D. SOFTWARE .....	19
V. OPERATIONAL SCENARIO .....	21
A. PRE-PASS .....	21
B. PASS .....	22
C. POST-PASS .....	23
VI. PCL: THE DETAILS .....	25

A. COMMAND ACKNOWLEDGEMENT .....	25
B. COMMAND VERIFICATION .....	26
C. DATA INTEGRITY .....	28
D. MULTIPLE COMMANDS PER DATA PACKET .....	29
E. SUPER USER/MISSION CRITICAL COMMANDS .....	31
F. PASSWORDS .....	31
G. VERBOSE USER SYNTAX .....	33
H. GROUND SIMULATION .....	34
VII. CONCLUSIONS AND FOLLOW ON WORK .....	35
APPENDIX A. PANSAT COMMAND LANGUAGE (PCL) .....	37
APPENDIX B. I/O PORTS AND THEIR DEFINITIONS .....	139
APPENDIX C. ADDRESSES AND SUBADDRESSES FOR PERIPHERAL DEVICES. ....	141
LIST OF REFERENCES .....	143
INITIAL DISTRIBUTION LIST .....	145

## **I. INTRODUCTION**

The Petite Amateur Navy Satellite (PANSAT) is a project sponsored and funded by the Navy Space Division, N-63 and administered by the Space Systems Academic Group (SSAG) of the Naval Postgraduate School (NPS). The military objectives of PANSAT are to enhance the education of military officers in the NPS Space Systems curricula and to demonstrate the feasibility of a quick-reaction, low cost communications satellite with digital, over-the-horizon communication, low probability-of-intercept and low probability-of-jamming characteristics. The educational objectives of PANSAT are to provide practical thesis research topics in space system engineering and operations, provide actual hardware and software design and integration experience, and in general, to give students exposure to development and life cycle issues associated with space systems.

### **A. PANSAT MISSION DESCRIPTION**

PANSAT will be a low-earth orbiting, free-tumbling satellite providing digital, store-and-forward communication using primarily spread spectrum modulation. Binary phase shift keying (BPSK) modulation is also available and may be selected by the ground station in lieu of spread spectrum modulation. The satellite structure is constructed of aluminum in a polyhedral configuration with a diameter of about 19 inches (Fig. 1). PANSAT may be launched from a Shuttle Get Away Special (GAS) canister, which provides a typical orbit of 330 km at 28.5° degree inclination. An average of six minutes of communication per pass is possible with this orbit, providing communication possibilities throughout the day. [Ref. 1]

General users on earth will be able to connect with PANSAT to send and receive electronic mail that is stored on-board, upload and download files, and read spacecraft telemetry. Multiple users will have the capability of communicating with PANSAT

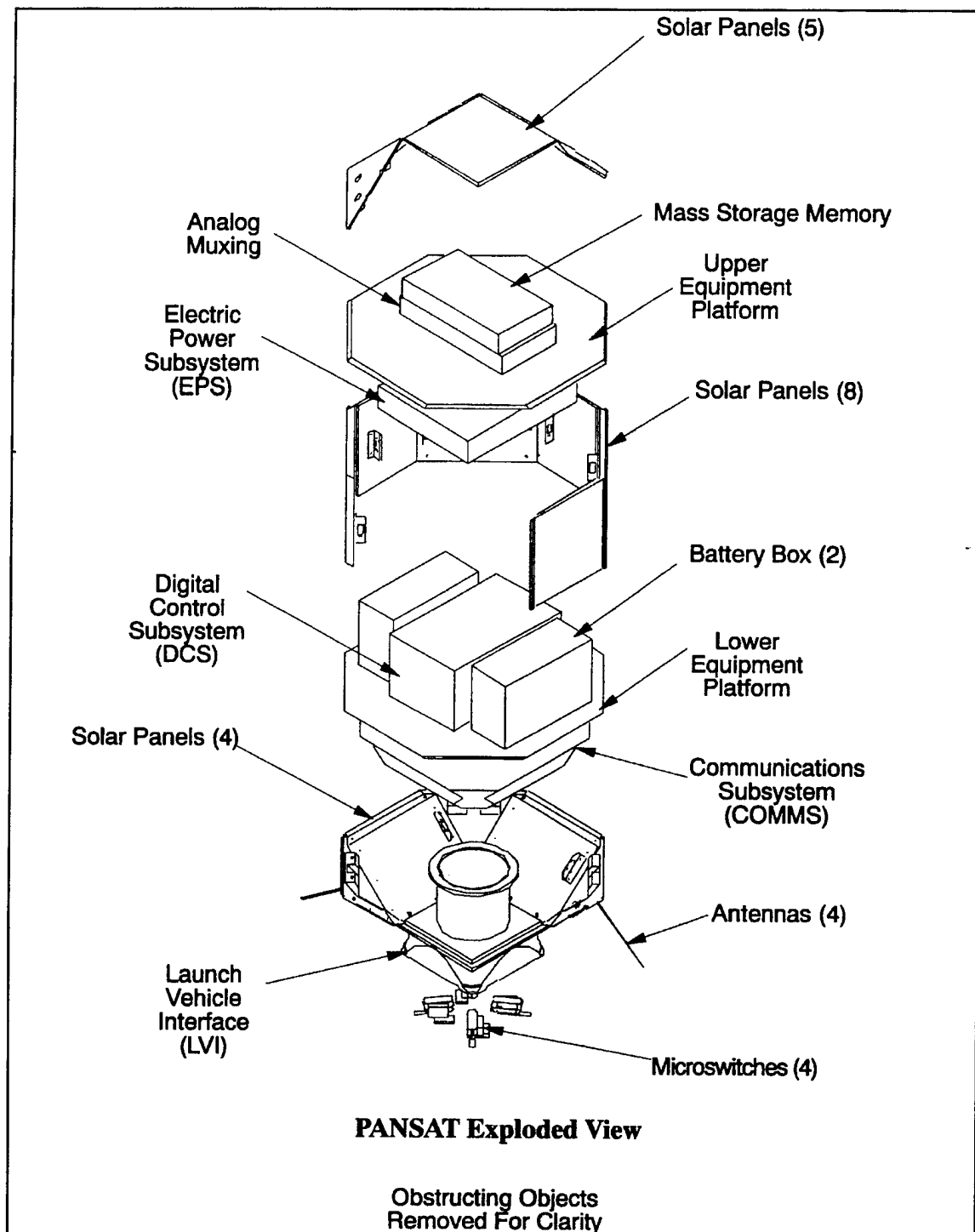


Fig. 1 PANSAT Exploded View

during the same window of time over one physical communication channel by using packet radio and the AX.25 protocol. [Ref. 1]

The NPS ground station will act as the Spacecraft Operations Control Center (SOCC) and will be used to perform satellite commanding, telemetry collection and software updates. [Ref. 1]

## **B. PANSAT COMMAND LANGUAGE**

The PANSAT Command Language (PCL) is used by the ground station to communicate or drive the Digital Control Subsystem (DCS) on board PANSAT. The DCS is PANSAT's command and data handling system. The DCS performs two major functions. It receives, validates, decodes and distributes commands to other spacecraft systems as well as gathers, processes and formats housekeeping and mission data for downlink, or use by an on-board processor. The DCS also performs additional functions, such as security interfaces, spacecraft time keeping, and computer health monitoring (watchdog). Hence, for the successful operation of PANSAT, an efficient and reliable means of communications with the DCS must be provided. The PCL is the vehicle/tool/method that provides such communications. [Ref. 2]

One requirement of the PCL is reliability. To date (Sept. 95), according to Space Systems Academic Group documents, the total projected life cycle cost of PANSAT is in excess of \$2 million dollars; hence, to have a catastrophic failure of the satellite because the commanding was unreliable, is unacceptable. In fact, any human induced catastrophic event would probably be considered unacceptable, except arguably for its educational benefit, which is one of PANSAT's primary missions.

In addition, based on the currently projected orbital parameters (51° inclination, eccentricity  $\approx 0$ , period  $\approx 92$  min.) the best case revisit scenario calculations show that PANSAT will be in view of the NPS ground station for an average of 6 minutes once,

possibly twice per day [Ref. 3]. With this limited time-in-view, communications become vital and are thus directly related to the efficiency and reliability of the PCL.

A number of ideas have been considered and implemented to ensure the reliability and efficiency of the PCL. These include: command acknowledgement, command verification, command encoding, single command per packet, verbose user syntax, super user commands, mission critical commands, and ground simulation.

## II. PANSAT HARDWARE ARCHITECTURE

The PANSAT hardware architecture is comprised of the electrical power subsystem (EPS), digital control subsystem (DCS) and the communications subsystem (Fig. 2). Incoming signals are received by the antenna and passed to the RF section. The RF section downconverts the signal from the UHF 436.5 MHz frequency to the 70 MHz intermediate frequency (IF). From the RF section the signal is passed to the DCS where they are further processed and made ready for use by other subsystems on board the satellite (Fig. 3).

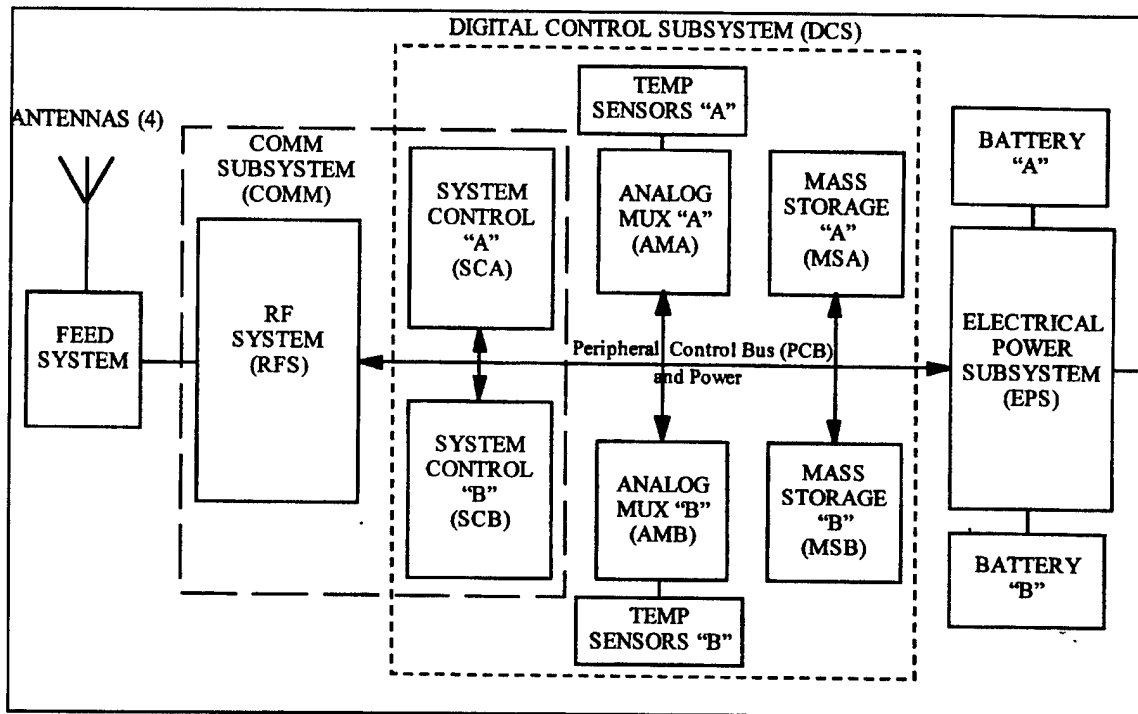


Fig. 2 PANSAT Hardware Architecture



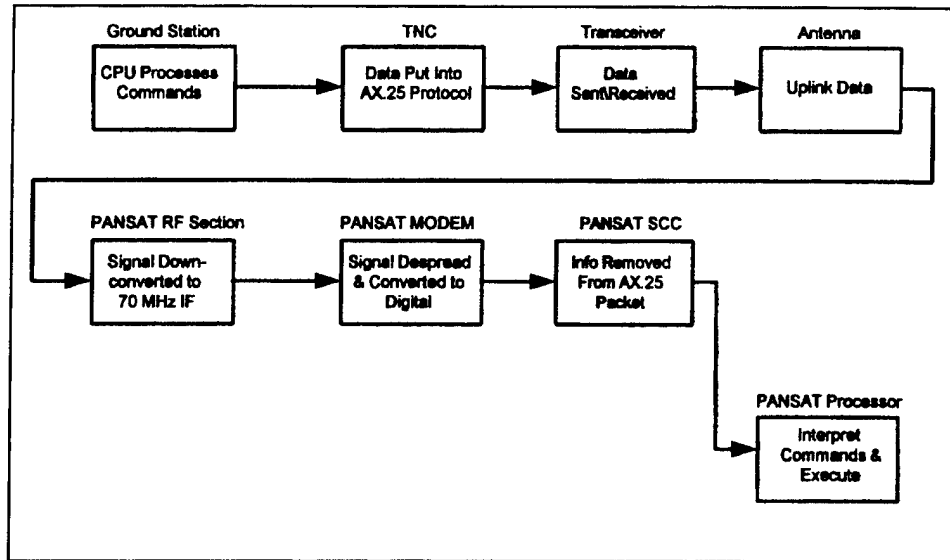


Fig. 3 Information Flow Diagram

## A. RF SUBSYSTEM

The basic components of the RF subsystem are two low noise amplifiers (LNA), two high pass amplifiers (HPA), two mixers, and nine switches of various types to control the system (Fig. 4). These components are combined to form dual transceivers.

It is anticipated that ground controllers will take advantage of every opportunity to communicate with PANSAT, but this is not a requirement. Due to a number of circumstances, such as holidays or natural disasters, it may be several days between communications sessions. PANSAT will however, be expecting daily interaction with the NPS ground station. If PANSAT does not communicate with the NPS ground station for a few days (the exact number of days is still to be determined), software on board the satellite will attempt to reconfigure the state of the RF subsystem, the idea being that one or more components of the RF subsystem may have failed and is thus preventing communications with the ground. The failure may or may not be in the RF subsystem, but if it is, cycling through the various switch settings and selecting different

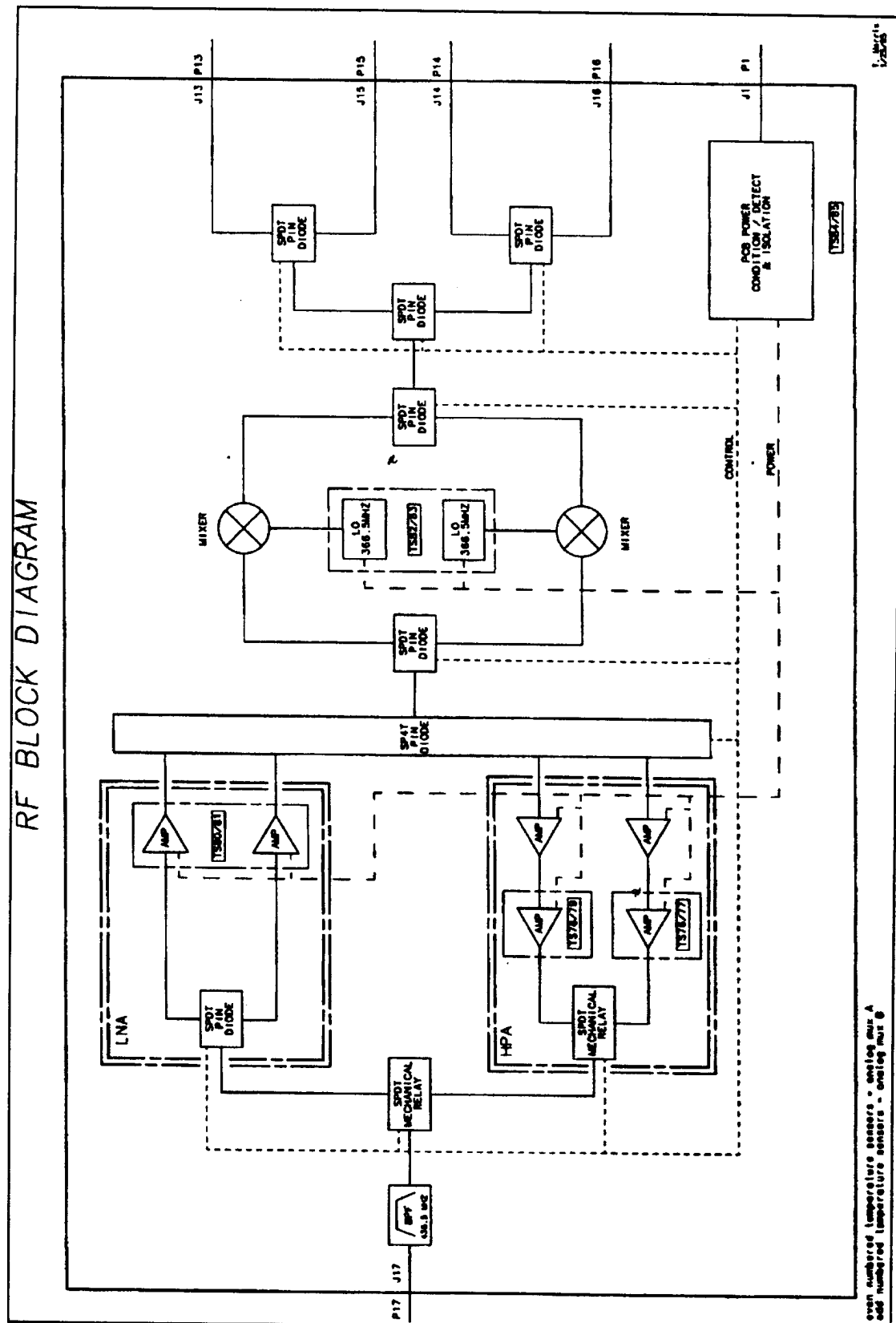


Fig. 4 PANSAT RF Block Diagram

combinations of components may solve the problem and allow communications to be reestablished. There are also high and low level PCL commands that may be issued to control the RF subsystem. So, in total there are three safety measures implemented to ensure reliability in the RF subsystem; the first being the fact that there are two transceivers, the second being that there will be software on board that will attempt to establish communication without ground intervention, and third being the PCL commands that can be used to exercise the system.

## **B. DIGITAL CONTROL SUBSYSTEM (DCS)**

The received signal next flows to the DCS where it is further demodulated and converted to a digital signal that may be used by the other subsystems. The primary functions of the DCS are to control and operate the RF subsystem, control the EPS, gather and store telemetry data, perform memory management, and control the message handling. The DCS basically controls everything on the satellite.

The DCS is fully redundant, consisting of two system control boards (SCA and SCB), two analog multiplexers (AMA and AMB), and two mass storage units (MSA and MSB) (Fig. 2). The analog multiplexers are used to collect temperature sensor data from throughout the spacecraft and send this data to the analog-to-digital (A/D) converters within the system controller. The mass storage units have a four megabyte capacity and are used for message and data storage. If a failure should occur on one of the DCS's paired components, such as MSA, a control board switching procedure is invoked to deactivate the failing components, in this case MSA and activate the redundant unit, in this case MSB. [Ref. 4]

Each of the system control boards (SCA and SCB) of the DCS have the following main components: A microprocessor, serial communications controller (SCC), analog-to-digital converter (A/D converter), programmable peripheral interface (PPI), a PA-100

spread spectrum demodulator board, error-detection-and-correction (EDAC) random access memory (RAM), and 32K of programmable read only memory (PROM) (Fig. 5).

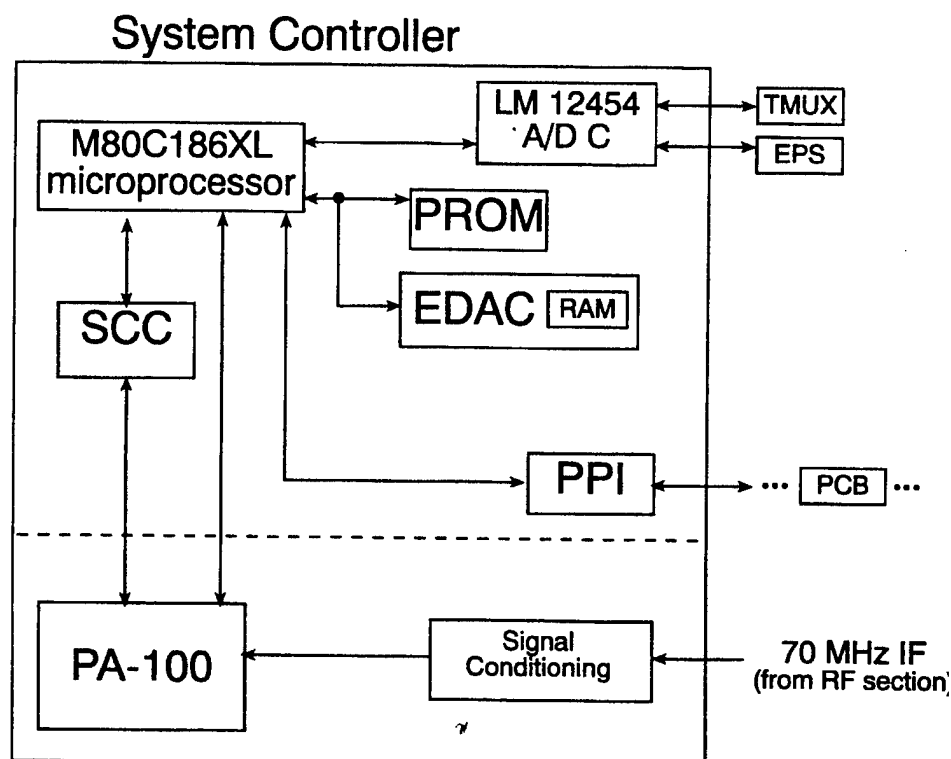


Fig. 5 PANSAT System Controller [After Ref. 4]

The microprocessor handles all of the software functions of the satellite. The SCC removes the data from the AX.25 protocol format and transfers digital information between the PA-100 and the microprocessor. The analog-to-digital converter interfaces directly with the microprocessor as an I/O addressed peripheral and accepts four separate analog inputs to provide one digital signal to the microprocessor. The PPI interface resides between the microprocessor and the peripheral control bus (PCB) and enables the system control boards to communicate with the other components connected to the PCB.

The MODEM board interfaces between the RF system and the SCC. It demodulates the incoming analog spread spectrum signal and converts it to a digital signal that may then be passed to and used by the microprocessor. The EDAC circuitry controls the SCB's RAM while the 32K PROM, which is connected directly to the microprocessors, contains the satellite initialization software. [Ref. 4]

The analog multiplexers gather data from the 60 temperature sensors, multiplexes the data together, and sends this data via dedicated cables to the A/D converter and then onto the microprocessor to be used accordingly. Each system controller can control either of the two analog multiplexers.

The two mass storage units connect to the microprocessor via the PPI and the PCB. Each unit has a four megabyte capacity and will provide storage for files, software, telemetry, and user mail. Each module also contains 512 kbits of flash memory. The flash memory will fly as an experimental, non-volatile memory device.

As with the RF system, there are a number of high and low level PCL commands that may be used to control the DCS and its components. So, not only is there hardware back-ups, but also software back-ups.

### **C. THE PCB AND THE PCL**

Although not considered a subsystem, the PCB is a critical component of the PANSAT hardware architecture. The PCB is the device that connects together and allows the DCS to communicate and control the satellite subsystems. The PCB is a comparatively simple and robust piece of hardware and for this reason, there is no back-up or redundant counterpart. The PCB does, however, constitute a single-point-of-failure for the satellite. Not only is the physical integrity of the PCB important, but being able to control the PCB is equally important.

At the most fundamental level the microprocessor is controlling everything that is happening on board the satellite. The microprocessor accomplishes this by sending and

receiving information via input/output ports (I/O ports). Dedicated I/O ports are used to physically connect the components of the System Control Boards. By using what are referred to as I/O read and write instructions, the microprocessor may either read data from, or write data to other System Control Board components thus, ultimately controlling the satellite. At the lowest level, the PCL allows ground controllers to issue satellite commands using I/O statements.

Above the microprocessor level, the PCB is controlling the satellite. Similar to I/O commands, ground controllers may use what are referred to as PCB read and write statements to control the PCB and again, ultimately the satellite. The distinction between I/O and PCB read and write statements is that from the user's perspective, PCB statements may be used to communicate directly between peripheral devices (via the PCB) without interaction with the microprocessor.

In reality though, PCB commands are derived from a series of I/O statements. For example, there is a known and finite number of combinations in which the system controller communicates with peripheral devices and the I/O statements to do so are known. Instead of issuing perhaps a dozen I/O commands to accomplish a specific and routine task such as reading telemetry, a single PCB command is issued and interpreted by the microprocessor, which in turn issues the required I/O commands.

## D. ELECTRICAL POWER SUBSYSTEM (EPS)

The primary function of the EPS is the collection, conversion, storage, and distribution of power to other subsystems in the satellite. The EPS consists of 18 solar panels, which are mounted to the surface of the satellite, two redundant nickel-cadmium (NiCd) batteries (BATTA and BATTB), voltage, current, and temperature sensors, a watchdog timer, power regulation and conditioning circuitry, and a launch start-up switch (Fig. 6). The EPS performs battery charge regulation (trickle, reconditioning, charge),

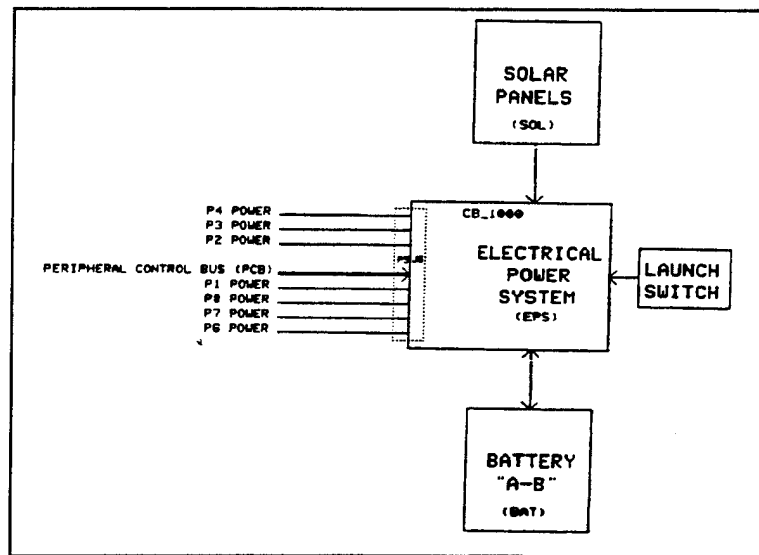


Fig. 6 PANSAT EPS [From Ref. 4]

multiplexes current and voltage readings, and gathers internal temperature data. Power is supplied by the EPS to other subsystems using dedicated circuitry on the PCB, and is controlled by the DCS. Current and voltage readings are collected in the EPS, multiplexed, and sent to the A/D converter located in the DCS. Temperature sensor lines from the EPS are sent to the temperature multiplexer modules. Readings from the DCS's

A/D converters are sent to the microprocessor, where decisions are made regarding the regulation and control of the EPS. Based on these decisions, the DCS sends instructions back to the EPS via the PCB, and the desired EPS configuration is set (Fig. 7). [Ref. 4]

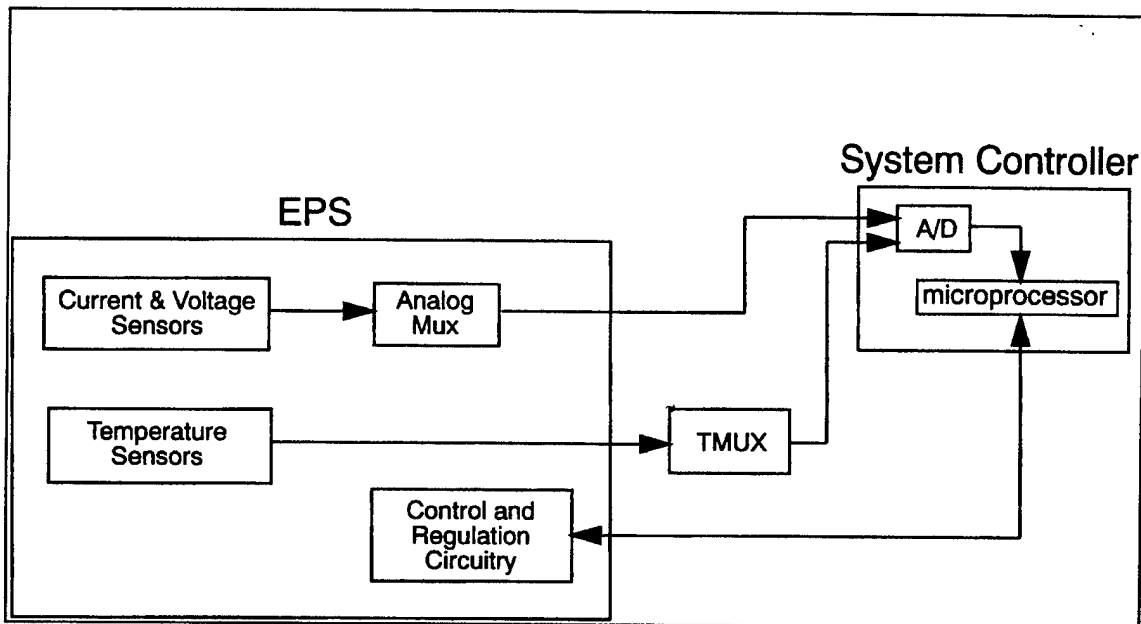


Fig. 7 EPS Signal Flow [After Ref. 4]

The EPS has 18 solar panels and two batteries, and while not optimal, the satellite may continue to operate using a reduced number of panels or a single battery. The control and regulation circuitry autonomously maintains the EPS within operating limits, and high and low level PCL commands may be issued to control the system as well. The EPS control and power switching boards are not redundant however. Failure of one of these boards may lead to the loss of the satellite.





### III. TELEMETRY

PANSAT telemetry may be thought of in two broad categories; hardware telemetry and software telemetry. There are approximately 94 hardware telemetry points. These include temperature data for each of the 18 solar panels and voltage data for each of the ten cells of each battery.

There are approximately 85 software telemetry points. Software telemetry provides ground controllers with statistics on the various events involving the satellite. These events include: number of logins and logouts, number of unauthorized login attempts, number of transmitted and received frames, and the number of errors in transmission. A complete listing of hardware and software telemetry points may be found in the PANSAT Software Detailed Design Document [Ref. 5].

Telemetry information enables ground controllers to evaluate PANSAT performance and gather health and status information. Additionally, telemetry data is continuously evaluated by the DCS to ensure that the other subsystems of the satellite are operating properly. If the DCS determines from the telemetry that a particular subsystem has malfunctioned or is approaching maximum or minimum operating conditions, the DCS will be able, to a limited extent, take autonomous corrective or preventative action to place the satellite in a safe state.

Telemetry data is stored in two separate locations on-board the satellite. A complete set of telemetry is written to both the mass storage and the flash memory. The rate at which telemetry is gathered and stored may be adjusted using PCL commands. High, medium and low level PCL commands may be used to instruct PANSAT to downlink its telemetry data. At the lowest level, I/O or PCB read commands may be issued to recover telemetry. This is not the easiest or preferred method of recovering telemetry, but nonetheless, it is possible. At the mid-level, PCL commands may be issued to recover specific blocks of SRAM or flash memory. These mid-level PCL commands were not intended to be used to recover telemetry, but like the low-level I/O

and PCB commands, reading specific storage locations to recover telemetry is a viable option. At the highest level, there are PCL commands to be used specifically for the recovery of telemetry. Like all PCL commands, these high level telemetry recovery commands are in actuality low-level I/O and PCB commands that are grouped together to form macros.

A final category of telemetry is known as recent telemetry. The DCS is continuously polling or gathering telemetry data. As mentioned above, the rate at which telemetry is collected and stored may be adjusted by issuing PCL commands. Recent telemetry is just that: the most recent value collected by a particular hardware sensor or software monitoring subroutine. Recent telemetry gives ground controllers the most up-to-date information on the status of PANSAT. How timely recent telemetry is, depends upon how often telemetry sensors gather information. If the telemetry polling rates are set at five minute intervals, ground controllers will know to an accuracy of five minutes the state of PANSAT. In practice, most telemetry data will be gathered on the order of seconds vice minutes and it should be kept in mind that the rate at which telemetry is gathered may be adjusted for each of the individual telemetry sensors.

## **IV. GROUND STATION HARDWARE AND SOFTWARE ARCHITECTURE**

The necessary components for the PANSAT ground station have been acquired and/or are being assembled by the Space Systems Academic Group and are located in Bullard Hall on the campus of the Naval Postgraduate School. The ground station will be used to perform a variety of tasks including commanding the satellite subsystems, uploading new software, and maintaining the store and forward messaging system. The ground station may be thought of as four distinct subsystems consisting of the transceiver, computer, antenna, and software.

### **A. TRANSCEIVER SUBSYSTEM**

The transceiver subsystem consists of a transceiver, terminal node controller (TNC) and a MODEM. The transceiver is an ICOM 970H multi-band all mode transceiver, capable of operating on the two meter amateur band (144-148 MHz) and the 70 centimeter amateur band (430-450 MHz). The radio can be controlled manually or by a computer (see Computer Subsystem description below). [Ref. 6]

The TNC is an Advanced Electronics Applications (AEA) Inc., model DSP-2232 multi-mode data controller. It interfaces between the computer and the transceiver and converts binary data from the computer to Frequency Shift Keying (FSK) modulated data for use by the radio, and vice versa. The DSP-2232 utilizes digital signal processing technology and is easily upgradable. [Ref. 6]

The MODEM used in the transmitter subsystem is specific to PANSAT. Existing amateur radio satellites such as WO-18, DO-17, and AO-13 utilize a variety of modulation schemes such as 1200 bps AFSK packet, 1200 bps AFSK ASCII, and CW to name a few. MODEMs for these modulation types are available to amateur radio operators from a number of vendors. PANSAT employs 9842 bps direct sequence spread spectrum modulation. There is currently no commercially manufactured MODEM

suitable for PANSAT. Therefore, members of the SSAG are designing and building such a MODEM. The specifications and design for this 9842 bps direct sequence spread spectrum MODEM will be made available to amateur radio operators, who may then build the MODEM themselves.

## **B. COMPUTER SUBSYSTEM**

The computer subsystem consists of a 486DX 33 MHz computer and an installed interface card called a Kansas City Tracker/Tuner (KCT). The KCT enables the computer, antenna rotor controller, and transceiver to communicate with each other. Utilizing satellite tracking software called Windows Satellite Programs (WiSP: see software subsystem below), the computer can point the antenna and tune the transceiver. This allows antenna pointing and radio tuning to be fully automated. [Ref. 6]

## **C. ANTENNA SUBSYSTEM**

The antenna subsystem consists of an antenna, a rotor, and a preamplifier. The antenna is a crossed yagi design and is manufactured by Cushcraft Corp. (model AOP-1). The crossed yagi design allows the antenna to transmit and receive circularly polarized signals from spin stabilized satellites. [Ref. 6]

The rotor is built by Yeasu Co. Ltd. (model G-5400B) and is actually a pair of rotors: One controlling the antenna azimuth (0-360 degrees) and the other controlling elevation (0-90 degrees). The KCT interfaces with the rotor controller and allows for computer control of the antenna system. [Ref. 6]

The preamplifier is manufactured by ICOM Inc. (model AG-45). It is most mounted near the antenna and used to amplify weak signals received from the satellite and send them to the transceiver. [Ref. 6]

## **D. SOFTWARE**

The PANSAT ground station will utilize three pieces of software, WiSP, the Graphical User Interface, and the PCL. The software that will control the antenna pointing and tracking is called WiSP and was written by Chris Jackson, ZL2TPO. WiSP is available from the American Satellite Corp. (AMSAT) and is widely used throughout the amateur radio community for PACKET satellite communications. WiSP is a compilation of seven separate programs designed to enable amateur radio operators to perform all the necessary functions to successfully communicate with PACSATs. These functions include among other things, antenna pointing and tracking, mail processing, and message creation and viewing.

WiSP was designed for users and not operators of amateur radio satellites. Therefore, a Graphical User Interface, which is tailored specifically for use with PANSAT, is being developed by Jens Bartschat, a German Naval Officer participating in the Department of the Navy's Personnel Exchange Program (PEP). WiSP will be run minimized in the windows environment and used only for antenna pointing and tracking and AX.25 protocol functions.

A Graphical User Interface specifically for PANSAT will be needed to interact with the spacecraft. Utilizing this interface, ground controllers will be able to access the PCL to compile and transmit command scripts. The Graphical User Interface will also be used to display telemetry data and to perform any required housekeeping functions.



## **V. OPERATIONAL SCENARIO**

Communication with satellites is typically thought of as occurring in three distinct phases consisting of pre-pass, pass, and post-pass events. This three phase methodology is particularly useful when communicating with low earth orbiting satellites such as PANSAT because it ensures maximum data throughput during the limited time in view.

### **A. PRE-PASS**

In many respects, the pre-pass phase of a communications session is the most critical. The amount and quality of preparation for a pending pass will directly affect the success of the pass and post-pass events and ultimately the satellite operation. During the pre-pass phase, ground controllers should verify the status of the ground station. This consists of ensuring that all of the equipment from the antenna to the transceiver is functioning properly, and if it is discovered that a piece of equipment has for some reason become inoperable, it should be repaired or replaced prior to the pass.

Ground controllers should review the operations log and the planned activities to familiarize themselves with the anticipated state of PANSAT and to be appropriately prepared for the upcoming pass. Ground station displays should also be activated and made ready for the pass.

The most important part of the pre-pass event is the construction of any command scripts that are to be transmitted during the pass. A command script consists of one or more individual commands selected from the PCL, that are grouped together and transmitted so as to command PANSAT into the desired state. Say for example that it has been determined that based on the telemetry from the previous pass that the mass storage was full and the system clock was inaccurate. To rectify the situation, a ground controller could choose the appropriate commands from the PCL and compile a script to perform the necessary tasks automatically once the communications link is established and without any additional interaction.



Scripting is a very powerful methodology and maximizes data throughput. Ground controllers are not sitting at a terminal issuing PCL commands manually, one by one, wasting critical time-in-view. Scripting also increases reliability because scripts may be compiled and simulated prior to a pass to ensure they have the desired effect on the satellite.

## **B. PASS**

The FCC prohibits satellite links being established less than 10 degrees above the horizon [Ref. 3]. Therefore, the pass phase begins when the satellite moves 10 degrees above the horizon and is in view of the ground station. WiSP is continuously calculating pointing and tracking data and sending this information to the computer subsystem, which in turn is commanding the antenna and transceiver subsystems in an attempt to establish the communications link.

Once the communications link is established, PANSAT will immediately be commanded to transmit its current telemetry data. Once this telemetry data is received, it will automatically be evaluated to see if the spacecraft is operating nominally. Ground controllers will have assembled command scripts based on nominal operating conditions and if this is not the case, the scripts that they have assembled may be inappropriate.

Once the state of PANSAT has been verified normal communication may begin. PANSAT will be commanded to complete downlinking any partially transmitted data or files from the previous pass such as events logs, telemetry, or command acknowledgements, and then begin sending new data. Ground controllers will also be able to uplink files, software, and perform housekeeping functions.

## C. POST-PASS

When PANSAT moves over the horizon and out of view of the ground station, the pass is over and the post-pass phase begins. At this point, everything that occurred during the pass should be logged in one way or another, either manually or automatically by the ground station computer.

The operations log, which will probably be hand written initially and entered into a data base later, should indicate at a minimum the time and date that the pass started and stopped, the primary ground controller's name and if the planned activities for the pass were accomplished. These actions may include checks to verify that anticipated data such as event logs and command acknowledgements were received. Also, any mail or files addressed to the ground station should be evaluated and distributed accordingly and software and hardware problems should be annotated.

It is envisioned that much of the record keeping, archiving, and report generating will be automated using the graphical user interface and that the requirement for hand written paper logs will be a minimum. All data that is transmitted to and received from PANSAT will pass through the graphical user interface, thus making the task of record keeping a straight forward process. For example, when the current telemetry data is received from PANSAT it will be interpreted and displayed by the graphical user interface. At the same time, transparent to the ground controller, this incoming telemetry will be archived and a record made that the telemetry was received. The same principle applies when uplinking to PANSAT. Any information which passes through the graphical user interface, which is essentially everything, is written to a file, thus greatly simplifying the task of record keeping, archiving and report generation.

Stored telemetry and ground controller logs will be used to identify any spacecraft anomalies or out-of-bound situations. Ideally, enough information will be available in the telemetry to determine the cause of such anomalies



## **VI. PCL: THE DETAILS**

To this point, the relevance of the satellite subsystems, PCB, telemetry, ground station, and operations as they relate to the PCL have been shown. To briefly summarize, the PCL may be used at the lowest level to control the microprocessor via I/O statements. At a higher level, the PCL may be used to control the subsystems via PCB statements. At a higher level still, the PCL may be used to control systems directly such as toggling between transceivers or mass storage units. Finally in the hierarchy of control, the PCL may be used for high-level control, such as retrieving telemetry. In this high-level control scenario, the PCL is not only controlling the mass storage unit directly, but instructing the mass storage to perform a specific task, namely, to determine where in the mass storage the telemetry is located, go there, retrieve the data and place this data on the PCB where it will be made ready by the other subsystems for transmission. The PCL is also an integral part of spacecraft operations and is a major consideration in the design and performance of the ground station. What follows, is a compilation of issues, thoughts, and ideas concerning the DCS of PANSAT. What follows are the details of the of the PCL.

### **A. COMMAND ACKNOWLEDGEMENT**

Command acknowledgement is a technique used to increase the PCL efficiency. In order to maximize communications during a given pass, PANSAT will not acknowledge every command from the PCL (and hence the ground station). Instead, only specific commands will be acknowledged (see Appendix A) and it should be assumed, that if a command was issued, it was executed. PANSAT will maintain an event log that will log all of the commands that it has executed. Therefore, if there is doubt as to whether or not a specific command was executed, say from out of limits telemetry data or an unexpected satellite state, the event log may be downlinked by ground controllers and analyzed for the appropriate indications. By assuming that issued

commands were executed properly, valuable time is not wasted informing ground controllers at the beginning of every communication session of the commands that have been executed since the last contact with PANSAT and thus, the available time-in-view of the satellite has been maximized.

PCL commands that will be acknowledged are those commands that require PANSAT to respond with some sort of data. For example, when ground controllers command PANSAT to downlink its current telemetry data, receipt of that data should serve as the acknowledgement that the command was executed properly. This technique increases efficiency by not requiring separate dedicated responses for both command acknowledgement and transmitted data. Instead, the received data is the acknowledgement.

## **B. COMMAND VERIFICATION**

Command verification differs from command acknowledgement in that command verification requires a response from the ground controller, while command acknowledgement involves a response from PANSAT. Also, command verification is an attempt to increase reliability, while as stated above, command acknowledgement is an attempt to increase efficiency.

Command verification is invoked automatically by the ground station software any time a ground controller attempts to issue a mission critical command, (see Appendix A). When a mission critical command is issued or a command script is built containing a mission critical command, the ground station software will recognize this and query the intentions of the operator. This gives the ground controller the opportunity to reflect for a moment on what their intentions really are. Was this simply a typo in the command script and was not really their intention at all, or after thinking about it, should they consult somebody before issuing this command? In any event, command verification is simply intended to provide another level of reliability to the PCL.

One attribute of command verification is, that like command acknowledgement, it is adjustable. That is, the PCL commands that trigger the command verification subroutine on the ground station software may be changed as the experience level of operators increases. Initially it is envisioned that command verification will be used only in conjunction with mission critical commands. Eventually ground controllers may be given a rating based on experience. Each rating will require a subset of commands to pass through the command verification algorithm. This gives new operators the opportunity to gain familiarity with the PCL while minimizing the probability that they are allowed to make a mission critical error. At the same time however, users that are very familiar with the PCL will not be constantly queried. And finally, while not recommended, it is even possible to disable command verification entirely.

Another scenario may be that after gaining operational experience with PANSAT it is determined that certain commands require an inordinate amount of time to execute during a given pass. Under these circumstances, it may be desirable to filter these commands through the command verification subroutine. For example, if an operator desires to downlink two days worth of telemetry and uplink a command script containing 20 instructions, this is a relatively large amount of data and there may not be enough time to accomplish this task. Command verification could be used in this scenario to notify the ground controller that the time-in-view is limited during this particular pass and it is recommended that the telemetry request be reconsidered in order to ensure the 20 instruction command script is transmitted.

## C. DATA INTEGRITY

A very powerful and easily implemented method of ensuring the reliability of the PCL is the use of the AX.25 protocol itself. The AX.25 protocol incorporates a technique known as a cyclic redundancy check (CRC-16) and can be explained as follows. Given a  $k$ -bit message, the transmitter generates an  $n$ -bit sequence, known as a cyclic redundancy check (CRC), so that the resulting frame, consisting of  $k+n$  bits, is exactly divisible by some predetermined number. The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error [Ref. 7]. It can be shown [Ref. 7, 8], with some qualification, that CRC-16 is capable of detecting all single-bit errors, all double-bit errors, and any odd number of errors. CRC-16 is a specific type of CRC and differs from CRC-12 or CRC-32 in that it is popular for 8 bit characters and results in a 16 bit CRC.

Command encoding is an attempt to increase both reliability and efficiency. Command encoding can be accomplished using any number of bits, so long as there are enough bits to provide for the number of commands needed. For example, if the PCL contained 25 commands (vice the actual number of approximately 50), at least five bits would be required ( $2^5 = 32$ ) for the command encoding because any number of bits less than five (ie., 4 bits, or  $2^4 = 16$ ) would not provide enough unique bit patterns to encode all 25 of the commands.

Specifically, eight bit command encoding was chosen because it provides the required number of unique bit patterns for the PCL, is efficient and works well with CRC-16 as discussed above. If 16 bits had been chosen instead of eight, the eight most significant bits of each pattern would not be required for uniqueness, but would still be transmitted, thus effectively slowing the data rate and leading to inefficiency. In addition, eight bit encoding is widely used for programming and leads to simplicity in terms of the actual coding of the communication software. Lastly, eight bit encoding makes possible another layer of reliability.

With eight bit encoding (or n bit encoding in general), it is conceivable that a packet may experience a burst error during transmission that is not detected by the CRC-16, resulting in a single bit being "flipped." If 11111111 is transmitted from the ground station and 11111110 is received, PANSAT may interpret this as a valid command and execute the command accordingly. However, this was not the intention of the ground station and depending on the nature of the command, could be potentially catastrophic. One technique for solving this bit flip problem is to formulate the bit patterns of the individual commands so that the flipping of any single bit will result in an invalid bit pattern. So now, when 11111111 is transmitted from the ground station and 11111110 is received, this is an invalid command and PANSAT disregards it. Eight bit encoding provides enough bit patterns to implement this technique.

#### **D. MULTIPLE COMMANDS PER DATA PACKET**

The transmission of multiple commands per data packet has an effect on both reliability and efficiency. If several commands are grouped together and transmitted in a single data packet, the probability of the packet being corrupted during transmission is increased. Simply put, the more data there is to corrupt, the greater the probability that the data will be corrupted.

So, trade-offs need to be made. If only one command is transmitted per AX.25 information field (see Fig. 8 for a description of the AX.25 information frame), PCL reliability is increased by minimizing the probability that a packet is corrupted, because there is a minimum of data to corrupt and the communications software is less complex. At the same time however, efficiency decreases because most of the transmitted information is overhead, not informational data.

Conversely, if an attempt is made to increase efficiency by sending multiple commands per data packet, reliability is reduced because the larger packet has a greater probability of being corrupted and the communications software is more complex.



Efficiency is decreased because more time is now being spent retransmitting corrupted frames rather than transmitting new frames.

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

Fig. 8 AX.25 Information Frame

Assuming the maximum information frame size of 332 bytes (2656 bits) and a bit error rate (BER) of  $1 \times 10^6$  bits/error, that means that there will be 0.0027 errors/frame, or 1 error in approximately every 370 frames with approximately 23% of the transmitted data as overhead information.

For comparison, now assume that only one command, that is 10 bytes in length, is transmitted in the AX.25 information field and that the frame size is now 86 bytes (688 bits) instead of 332 bytes. Using the same BER, there is now expected to be 1 error in approximately every 1454 frames, but 88% of the data is now overhead.

Another consideration is how PANSAT will process multiple commands per data packet. For instance, will PANSAT read the entire information field of the AX.25 frame and execute the commands within that field based on some sort of priority scheme, or will commands be executed as they are received, in a First In First Out (FIFO) fashion, regardless of their relative importance or, should each command be uplinked one at a time in its own command message? If PANSAT executes instructions as they are received, this increases the complexity of the dialog between the ground station and PANSAT. For example, if the spacecraft is processing a telemetry request command while uplink commands are still being sent, the possibility of packet collision increases.

Prioritization of PCL commands in uploaded scripts will be performed by the ground station software. Command scripts which require multiple downlinks such as telemetry or event logs, or time-sensitive processing will be divided on the ground into

separate command messages. These messages will be uplinked one at a time allowing time for the downlink or command execution. By doing this, the probability of transmission collisions, of downlinked telemetry and uplinked commands, is avoided.

## **E. SUPER USER/MISSION CRITICAL COMMANDS**

In order to increase reliability by restricting user access, certain commands of the PCL have been designated as superuser commands. Superuser commands are those commands that affect the configuration of the satellite (see Appendix A for a listing and description of Super User commands) and will require a valid password prior to transmission.

Mission Critical commands have also been designated (see Appendix A for a listing and description of Mission Critical commands) . Mission Critical commands are a subset of Super User commands and as such, also require a valid password prior to transmission. Like Super User commands, Mission Critical commands may also affect the state of PANSAT, but can do so irreversibly and catastrophically, so care should be taken when issuing Mission Critical commands.

## **F. PASSWORDS**

Passwords are an attempt to increase the reliability of the PCL. The majority of commercial and military satellites utilize sophisticated encryption algorithms to encrypt their telemetry data. Encryption offers security and hence reliability by ensuring that only authorized users are allowed to interact with the satellite.

PANSAT has no need for sophisticated encryption. Its primary mission is as a teaching and learning aid for the Space Systems Academic Group and not as a secure satellite communications system. Additionally, PANSAT will be made available to amateur radio operators around the globe and encrypting telemetry would be senseless.

There are however commands in the PCL that if issued improperly or maliciously, could cause catastrophic failure to PANSAT. It has been decided for this reason, that there should be some simplified or elementary security measure implemented. One security option considered is the use of passwords. Prior to the launch of PANSAT, a password table will be generated and a copy placed in the satellite's static ROM as well as on the NPS ground station computer. An individual password will consist of 7 randomly generated alpha numeric ASCII characters and there will be approximately 1600 unique passwords in the table (enough for the expected 2-3 year mission duration). Under normal circumstances, passwords will change once per day.

When the NPS ground station establishes communications with PANSAT, PANSAT will automatically transmit its current on-board clock information, that is, the time that PANSAT thinks it is. For a variety of reasons, this date and time information may or may not be accurate. Regardless of the accuracy, the ground station software will take this information, enter into the password table and retrieve the proper password for that particular date and time. Henceforth, a correct password will be appended automatically to any commands requiring one (see appendix A for a list of commands that require a password) without any interaction with the ground controller. This automated password look-up method provides an additional layer of security in that passwords will not be readily available and mistakes from manual look-ups will be eliminated. A hard copy printout of the password table will be kept locked in the SSAG safe in Bullard 125.

It is anticipated however, that for a variety of reasons, PANSAT will be resetting its systems and hence its on-board clock at frequent intervals. This means that when PANSAT transmits its date/time information at the beginning of a communications session, the password look-up subroutine will locate a password that has already been used, possibly dozens of times. Although unlikely, it is conceivable that a particularly motivated individual conducting TT&C analysis/surveillance may detect the fact that the same passwords are being used over and over, and use this observation to the detriment of the satellite. Therefore, the first three days of passwords (which is the longest anticipated

time that NPS will not communicate with PANSAT) will change every 30 minutes vice once per day. Hopefully, any ongoing surveillance effort will essentially be negated by changing passwords so frequently that packet recovery becomes impossible.

## **G. VERBOSE USER SYNTAX**

Verbose user syntax is an attempt to increase the efficiency and reliability of the PCL. Individual commands of the PCL could have been constructed with a variety of different syntaxes. For example, the syntax for the command to charge battery B could have looked like: *charge\_battery\_b*, or *chargebatteryb*, or *ch\_batt\_b*, or *etc*. Command syntax should be verbose enough to be easily understood and recognizable, but at the same time concise. When issuing commands or building a command script, a ground controller will essentially be using a text editor. Easily understood and recognizable commands increases the reliability of the PCL by reducing confusion and mistakes. At the same time, the concise nature of the commands eliminates unnecessary keystrokes and makes command editing more efficient.

It should be kept in mind that the actual length or number of characters associated with an individual command is only a consideration for editing and not in data throughput. When individual commands or command scripts are written with the ground station command editor, these ASCII character representations of the commands (or scripts) are converted, along with any required parameters, to a one byte command number and then to hexadecimal format prior to transmission. Therefore, even though an individual command may be 20 characters long, this does not mean that 20 bytes need be transmitted to issue the command. Instead, the 20 character long command will be converted to an equivalent one byte hexadecimal value and then transmitted.

## **H. GROUND SIMULATION**

Ground simulation is an attempt to increase efficiency and reliability. Based on the commands issued and telemetry received from the previous contact, ground controllers will be able to estimate (albeit not precisely) the state of PANSAT on the next pass. Utilizing these estimates, commands and command scripts may be prepared ahead of time, simulated on the ground, and made ready for transmission the moment communications are established with PANSAT. Ground simulation increases reliability by ensuring that the commands to be issued have their intended response. Efficiency is increased by not utilizing the limited time-in-view sending commands which are determined inappropriate through ground-based simulation.

## **VII. CONCLUSIONS AND FOLLOW ON WORK**

This thesis documents the thoughts and ideas various people of the Space Systems Academic Group have had on PANSAT command and data handling and results in the first iteration of the PCL. The PCL incorporates a number of techniques that increase efficiency and reliability and is therefore very robust and ideal for the military and educational missions of the satellite.

As the launch of PANSAT draws closer, additional iterations need to be performed on the PCL in order to further refine reliability and efficiency. Already, additional required commands and opportunities to combine existing commands have been identified. Also, there are still a number of PANSAT response data formats such as telemetry data, that are undefined and need to be determined. Finally, and it has already begun, the PCL needs to be translated from this document and made into compiled usable software, which in turn will need to go through several iterations for a final reliable product.



## **APPENDIX A. PANSAT COMMAND LANGUAGE (PCL)**

Appendix A, which is intended to be used as a reference source by ground controllers and software designers, is a detailed description of the PCL, and specifies among other things user syntax, command descriptions and expected responses to commands. The Appendix is arranged in alphabetical order by command name.



## TIME-TAGGED COMMANDING

Operation: Add a command from the PANSAT command buffer

Command: **add\_cmd** <Command> <Time>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>add_cmd</b>	Password (4 Bytes)	Command (1 Byte) <Command>	Parameters (1 Byte) <Parameters>	UTC (4 Bytes) <Time>
------------------------------------	-----------------------	-------------------------------	-------------------------------------	-------------------------

AX.25 Information Field

Required Parameters:   <Command> (A valid PANSAT command)  
                          <Time>       (dd/mm/yy/hh/mm/ss)

Optional Parameters:   None

Function:               **add\_cmd** <Command> <Time> is used to add a command to the PANSAT command buffer. The required parameters are the <Command> to add to the buffer and the <Time> that the command should be executed. Upon receipt, PANSAT will verify that the <Time> parameter is valid and will ensure that no two commands have the exact same time of execution. If

PANSAT determines that two commands have the same time of execution, the commands will be prioritized on the basis of which command was sent to the buffer first. The first command to the buffer will be executed first (FIFO). The number of commands allowed in the command buffer is only limited by the size of the buffer.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **add\_cmd <Command> <Time>** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was added to the command buffer or that the command was executed at the specified time, the ground controllers should analyze the event log for the appropriate indications.

## TASK CONTROL

Operation: Upload a software task to PANSAT

Command: **add\_task** <Task Name>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>add_task</b>	Password (4 Bytes)	Task Name <Task Name>	Task Length (2 Bytes)	File Data ... (x Bytes)
-------------------------------------	-----------------------	--------------------------	--------------------------	----------------------------

AX.25 Information Field

Required Parameters:     <Task Name> The name of the task to be uploaded and may be up to eight characters in length. All software tasks have the same three character extension. All software tasks to be added or uploaded will be located in a separate default software task subdirectory on the ground station hard drive and therefore a path need not be specified when issuing this command. The default software task subdirectory may however be overridden and specified by the ground controller if the need arises.

Optional Parameters:     None

Function:                **add\_task** <Task Name> is used to upload software tasks to PANSAT. Software tasks are executable programs that are written and compiled on the ground and instruct PANSAT how to perform under various circumstances.

Super User:             Yes

Expected Response:    PANSAT will not directly confirm or verify the command **add\_task** <Task Name> and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the task was added to the task queue, ground controllers should execute a **list\_task** command to verify the contents of the task queue and the status of each task (PANSAT is limited to a maximum of 30 tasks). Ground controllers may also examine the event log in order to confirm that the command was executed.

## OS CONTROL

Operation: Perform a ROM boot on currently selected DCS

Command: **boot\_rom**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>boot_rom</b>	Password (4 Bytes)
-------------------------------------	-----------------------

AX.25 Information Field

Required parameters: None

Optional Parameters: None

Function: **boot\_rom** is used to reset or "warm boot" the software on the current system controller. This forces a complete execution of all of the initialization code, including the battery charge monitor and wait for NPS contact followed by the software uploads from NPS. **boot\_rom** only initializes hardware, it does not power cycle the hardware.

Super User: Yes

Expected Response: Assuming no battery charging is necessary, PANSAT will be listening for NPS to begin software uploads. If charging is required, PANSAT will complete battery charging then proceed to listen for the NPS ground station.

## BATTERY CONTROL

Operation: Charge Battery A

Command: **charge\_batt\_a**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>charge_batt_a</b>	Password (4 Bytes)
--	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **charge\_batt\_a** commands PANSAT to charge battery A.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **charge\_batt\_a** and unless there are indications to the contrary,

it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log and telemetry data for the appropriate indications.



## BATTERY CONTROL

Operation: Charge Battery B

Command: **charge\_batt\_b**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>charge_batt_b</b>	Password (4 Bytes)
--	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **charge\_batt\_b** commands PANSAT to charge battery B.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **charge\_batt\_b** and unless there are indications to the contrary,

it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log and telemetry data for the appropriate indications.

## TIME-TAGGED COMMANDING

Operation: Delete a command from the PANSAT command buffer

Command: **delete\_cmd** <Command> <Time>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Frame 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>delete_cmd</b>	Password (4 Bytes)	Command (1 Byte) <Command>	UTC (4 Bytes) <Time>
---------------------------------------	-----------------------	-------------------------------	-------------------------

AX.25 Information Field

Required Parameters:   <Command> (A valid PANSAT command)  
                          <Time>       (dd/mm/yy/hh/mm/ss)

Optional Parameters:   None

Function:               **delete\_cmd** <Command> <Time> is used to delete a command from the PANSAT command buffer. The required parameters are the <Command> to delete from the buffer and the <Time> that the command was to have been executed.

These two parameters will be compared to the contents of the command buffer and the appropriate entry will be deleted.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **delete\_cmd** <Command> <Time> and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not a particular command was deleted from the command buffer, the ground controllers should analyze the event log for the appropriate indications.

## FILE SYSTEM CONTROL

Operation: Delete a file from PANSAT

Command: **delete\_file** <Filename>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>delete_file</b>	Password (4 Bytes)	DOS File Name <Filename>
--	-----------------------	-----------------------------

AX.25 Information Field

Required Parameters: <Filename> DOS filename format. May be up to eight characters in length followed by an optional three character extension and a null string.

Optional Parameters: Use of wildcards is still to be determined.

Function: The command **delete\_file** <Filename> is used to delete specific files from the file directory.

Super User: No

Expected response: PANSAT will not directly confirm or verify the command **delete\_file <Filename>** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, ground controllers should examine both the file directory and the event log for the appropriate indications.

## TASK CONTROL

Operation: Delete a software task from PANSAT

Command: **delete\_task** <Task Name>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>delete_task</b>	Password (4 Bytes)	Task Name <Task Name>
--	-----------------------	--------------------------

AX.25 Information Field

Required Parameters: <Task Name> The name of the task to be deleted from the PANSAT task queue. May be up to eight characters in length. All software tasks will have the same three character extension.

Optional Parameters: None

Function: **delete\_task** <Task Name> is used to delete a software task from the PANSAT task queue. Software tasks are executable programs that are written and compiled on the ground and instruct PANSAT how to perform under various circumstances.

**NOTE: A SOFTWARE TASK MUST BE STOPPED  
BEFORE IT CAN BE DELETED.**

Super User: Yes

Expected Response: Although not mission critical, PANSAT will query the NPS ground station prior to the execution of the command **delete\_task <Task Name>** in order to verify its intentions. The NPS ground station will be immediately notified that a software task has been deleted and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that a software task has been deleted, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communication sessions. At this point, if there is doubt whether or not the task was deleted from the task queue, ground controllers should execute a **list\_tasks** command to verify the contents of the task queue and the status of each task (PANSAT is limited to a maximum of 30 tasks). Ground controllers may also examine the event log in order to confirm that the command was executed.



## FILE SYSTEM CONTROL

Operation: Get the PANSAT file directory

Command: **directory**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>directory</b>
--------------------------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: The command **directory** is used to get a listing of the PANSAT file directory.

Super User: No

Expected response: PANSAT will not directly confirm or verify the command **directory**. Receipt of the directory should serve as the indication that the command was executed properly. If there is doubt whether or not the command was executed, ground controllers should examine the event log for the appropriate indications. The protocol syntax for the response to the command **directory** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) directory	Directory Length (2 Bytes) <Length>	Directory Contents ... (X Bytes)
-------------------------------	--	-------------------------------------

AX.25 Information Field

## BATTERY CONTROL

Operation: Discharge Battery A

Command: **discharge\_batt\_a**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>discharge_batt_a</b>	Password (4 Bytes)
---	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **discharge\_batt\_a** commands PANSAT to discharge battery A.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **discharge\_batt\_a** and unless there are indications to the

contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log and telemetry data for the appropriate indications.

## BATTERY CONTROL

Operation: Discharge Battery B

Command: **discharge\_batt\_b**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>discharge_batt_b</b>	Password (4 Bytes)
---	-----------------------

AX.25 Information field

Required Parameters: None

Optional Parameters: None

Function: **discharge\_batt\_b** commands PANSAT to discharge battery B.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **discharge\_batt\_b** and unless there are indications to the

contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log and telemetry data for the appropriate indications.

## USER CONTROL

Operation: Disconnect communications with all current users.

Command: **drop\_users**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>drop_users</b>	Password (4 Bytes)
---------------------------------------	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **drop\_users** commands PANSAT to discontinue communications with all current users except the NPS ground station. This command should be issued whenever it is necessary for the NPS ground station to have exclusive access to PANSAT for such purposes as software uploading or troubleshooting. For most scenarios the **drop\_users** command

will be used in conjunction with the **lockout\_users** command to ensure that once all unauthorized users have been disconnected, no users will be allowed to log onto PANSAT.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **drop\_users** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log for the appropriate indications.



## FILE SYSTEM CONTROL

Operation: Download a file from PANSAT

Command: **get\_file** <Filename>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>get_file</b>	File Name <Filename>
-------------------------------------	-------------------------

AX.25 Information Field

Required Parameters: <Filename> DOS filename format. May be up to eight characters in length followed by an optional three character extension and a null string.

Optional Parameters: None

Function: The command **get\_file** <Filename> is used to get or download a specified file from PANSAT.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **get\_file <Filename>**. Receipt of the specified file should serve as indication that the command was executed properly. If there is doubt whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **get\_file <Filename>** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>get_file</b>	File Name <b>&lt;filename&gt;</b>	File Length (4 Bytes)	File Data ... (XXX Bytes)
-------------------------------------	--------------------------------------	--------------------------	------------------------------

AX.25 Information Field

## DIRECT LOW-LEVEL OPERATIONS

Operation: Perform I/O port read

Command: **io\_read** <Port>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>io_read</b>	Port (2 Bytes) <b>&lt;Port&gt;</b>
------------------------------------	---------------------------------------

AX.25 Information Field

Required Parameters: **<Port>** The CPU I/O port to be read. Hex range of allowable values is 0-FFFF. A complete listing of I/O ports and their definitions can be found in the Appendix B.

Optional Parameters: None

Function: The **io\_read** command is used to directly read the I/O ports of the microprocessor. These ports must be controlled in order to communicate with various PANSAT peripheral devices such as

the programmable peripheral interface (PPI) and serial communications controller (SCC). Under normal operating conditions, I/O statements are embedded within the PANSAT Command Language (PCL) and are thus transparent to the ground controllers. I/O statements can be quite complicated, even to perform the simplest of tasks and this method of embedding the I/O statements simplifies PANSAT commanding and reduces the probability of errors.

Super User: No

Expected Response: **Care should be taken when issuing I/O read commands.** PANSAT will not directly confirm or verify individual I/O read commands. Receipt of the requested information should serve as the indication that the command was executed properly. Since it is difficult to anticipate every possible response from an I/O read command, and since the responses may be somewhat cryptic, it is recommended that a team of ground controllers be assembled prior to utilizing I/O statements, to thoroughly discuss expected responses and potential consequences associated with a specific I/O read command. The protocol syntax for the response to the command **io\_read** **<Port>** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>io_read</b>	Port (2 Bytes) <b>&lt;Port&gt;</b>	Data (1 Byte) <b>&lt;Data&gt;</b>
------------------------------------	---------------------------------------	--------------------------------------

AX.25 Information Field

Where **<Data>** is the 1 byte data value to be read from the specified **<Port>**. The hex range of allowable **<Data>** values is 0-FF.

## DIRECT LOW-LEVEL OPERATIONS

Operation: Perform I/O port write

Command: **io\_write** <Port> <Data>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>io_write</b>	Password (4 Bytes)	Port (1 Byte) <Port>	Data (1 Byte) <Data>
-------------------------------------	-----------------------	-------------------------	-------------------------

AX.25 Information Field

Required Parameters:     **<Port>** The CPU I/O port to be written to. Hex range of allowable values is 0-FFFF. A complete listing of I/O ports and their values can be found in Appendix B.

**<Data>** The 8 bit data value to be written to the port specified using <Port>. Hex range of allowable values are 0-FF.

Optional Parameters:     None

Function: The **io\_write** command is used to directly write to the I/O ports of the microprocessor. These ports must be controlled in order to communicate with various PANSAT peripheral devices such as the programmable peripheral interface (PPI) and serial communications controller (SCC). Under normal operating conditions, I/O statements are embedded within the PANSAT Command Language (PCL) and are thus transparent to the ground controllers. I/O statements can be quite complicated, even to perform the simplest of tasks and this method of embedding the I/O statements simplifies PANSAT commanding and reduces the probability of errors.

Super User: Yes

Expected Response: **Care should be taken when issuing I/O write statements.** PANSAT will not directly confirm or verify individual I/O write commands, including mission critical commands. since it is difficult to anticipate very possible response from an I/O write command and since the response may be somewhat cryptic, it is recommended that a team of ground controllers be assembled prior to utilizing I/O write statements, to thoroughly discuss expected responses and potential consequences associated with specific I/O write commands.

## TIME-TAGGED COMMANDING

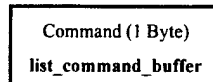
Operation: List the contents of the command buffer

Command: **list\_command\_buffer**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame



AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: The command **list\_command\_buffer** is used to display the contents of the command buffer.

Super User: No

Expected response: PANSAT will not directly confirm or verify the command **list\_command\_buffer**. Receipt of the command buffer should



serve as the indication that the command was executed properly. If there is doubt whether or not the command was executed, the ground controller should analyze the event log for the appropriate indications. The protocol syntax for the response to the command **list\_command\_buffer** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>list_command_buffer</b>	Length (2 Bytes) <Count>	Command Buffer Contents .... (XXX Bytes)
--	-----------------------------	---

AX.25 Information Field

## TASK CONTROL

Operation: List the software tasks currently on board PANSAT.

Command: **list\_tasks**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 1028 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	--------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>list_tasks</b>
---------------------------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **list\_tasks** is used to list all of the software tasks currently in the PANSAT task queue. Software tasks are executable programs that are written and compiled on the ground and instruct PANSAT how to perform under various circumstances.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **list\_tasks**. Receipt of the task list should serve as the indication that the command was executed properly. Ground controllers may also examine the event log in order to confirm that the command was executed. There may be a maximum of 30 tasks in the PANSAT task queue. Once the task queue is received from PANSAT, ground controllers will be presented with a graphical display showing the contents of the task queue and the status of each individual task. The protocol syntax for the response to the command **list\_tasks** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 1028 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	--------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>list_tasks</b>	Task List Length (1 Byte)	List of Task Names ... (X Bytes)
---------------------------------------	------------------------------	-------------------------------------

AX.25 Information Field

## USER CONTROL

Operation: Lockout new users.

Command: **lockout\_users**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>lockout_users</b>	Password (4 Bytes)
--	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **lockout\_users** restricts users from logging onto PANSAT. This command will most likely be used in conjunction with the **drop\_users** command to enable the NPS ground station to have exclusive access to PANSAT for purposes of uploading software or troubleshooting. **lockout\_users** may also be time tagged and used in conjunction with the **unlock\_users**

command in order to restrict the number of users during eclipse and hence reduce power consumption, or to prepare PANSAT for an impending communications session with the NPS ground station.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **lockout\_users** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log for the appropriate indications.

## DIRECT LOW-LEVEL OPERATIONS

Operation: Perform a peripheral control bus (PCB) read

Command: **pcbr** <Address>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

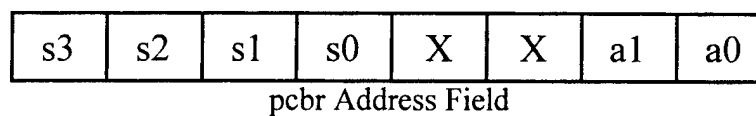
Command (1 Byte) <b>pcbr</b>	Address (1 Byte) <Address>
---------------------------------	-------------------------------

AX.25 Information Field

Required Parameters: **<Address>** The required **<Address>** parameter is one byte in length and specifies the PCB address and sub-address to be read. The four most significant bits of the address byte specifies the peripheral device to be read, while the two least significant bits specify the sub-address of the selected peripheral device. A complete listing of addresses and sub-addresses for the peripheral devices can be found in Appendix C.

Optional Parameters:     None

An example of the address field for use with the **pcbr** <Address> command. The four most significant bits (s3-s0) specify the address of the peripheral device to be read, while the least two significant bits (a1-a0) specify the sub-address of the selected peripheral device. Bit positions three and four are not used.



**Function:**                      Peripheral control bus (PCB) read/write statements are the lowest, most basic level of the PANSAT command language (PCL). Every command in the PCL is made up of a series of PCB read/write statement bundled together to form a single macro-type command. Thus, instead of having to issue several PCB read/write commands to perform a specific task, a single command is issued that incorporates all of the required PCB read/write commands for that specific task. This macro-type commanding simplifies PANSAT telemetry tracking and commanding (TT&C) and greatly reduces the probability of user error. It is anticipated that PCB read and write commands will be used only occasionally, or during troubleshooting, thus simplifying the effort by essentially bypassing the PCL and communicating directly with the peripheral devices.

Super User: No

Expected Response: **Care should be taken when issuing PCB read commands.**

PANSAT will not directly confirm or verify individual PCB read commands. Receipt of the requested information should serve as the indication that the command was executed properly. Since it is difficult to anticipate every possible response from a PCB read command and since the responses may be somewhat cryptic, it is recommended that a team of ground controllers be assembled prior to utilizing PCB statements, to thoroughly discuss expected responses and potential consequences associated with a specific PCB read statement. The protocol syntax for the response to the command **pcbr <Address>** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>pcbr</b>	Address (2 Bytes) <b>&lt;Address&gt;</b>	Data (1 Byte) <b>&lt;Data&gt;</b>
---------------------------------	---	--------------------------------------

AX.25 Information Field

Where **<Data>** is the 1 byte data value to be read from the specified **<Address>**. The hex range of allowable **<Data>** values is 0-FF.



## DIRECT LOW-LEVEL OPERATIONS

Operation: Perform a peripheral control bus (PCB) write

Command: **pcbw** <Address> <Data>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information frame

Command (1 Byte) <b>pcbw</b>	Password (4 Bytes)	Address (1 Byte) <Address>	Data (1 Byte) <Data>
---------------------------------	-----------------------	-------------------------------	-------------------------

AX.25 Information field

Required Parameters:     **<Address>** The required **<Address>** parameter is one byte in length. The four most significant bits specify the addresses of the various peripheral devices, while the four least significant bits specify the sub-addresses of the selected peripheral device. A complete listing of addresses and sub-addresses for the peripheral devices can be found in Appendix C.

Optional Parameters:     None

Example:

s3	s2	s1	s0	X	X	a1	a0
----	----	----	----	---	---	----	----

Address Field

Function: Peripheral control bus (PCB) read/write statements are the lowest, most basic level of the PANSAT command language (PCL). Every command in the PCL is made up of a series of PCB read/write statements bundled together to form a single macro-type command. Thus, instead of having to issue several PCB read/write commands to perform a specific task, a single command is issued that incorporates all of the required PCB read/write commands for that specific task. This macro-type commanding simplifies PANSAT TT&C and greatly reduces the probability of user error. It is anticipated that PCB read and write commands will be used only occasionally, or during troubleshooting, thus simplifying the effort by essentially bypassing the PCL and communicating directly with the peripheral devices.

Super User: Yes

Expected Response: **Care should be taken when issuing PCB write commands.**  
PANSAT will not directly confirm or verify individual PCB

write commands, including mission critical commands. Since it is difficult to anticipate every possible response from a PCB write command and since the response may be somewhat cryptic, it is recommended that a team of ground controllers be assembled prior to utilizing PCB statements, to thoroughly discuss expected responses and potential consequence associated with a specific PCB write statement.

## TIME-TAGGED COMMANDING

Operation: Purge the entire PANSAT command buffer

Command: **purge\_command\_buffer**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>purge_command_buffer</b>	Password (4 Bytes)
---	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **purge\_command\_buffer** is used to purge the current contents of the command buffer.

Super User: Yes

Expected response:

**purge\_command\_buffer** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the command buffer has been purged and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the command buffer has been purged, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indications that the ground controllers will have that the command buffer was purged will be in the event log.

## EVENT LOG

Operation: Purge the current event log.

Command: **purge\_event\_log**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>purge_event_log</b>	Password (4 Bytes)
--	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **purge\_event\_log** instructs PANSAT to delete the contents of the event log. The event log may be purged at any time, regardless of whether or not it has been downlinked.

Super User: Yes

Expected Response: **purge\_event\_log** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the log has been purged and an entry will be made to the new event log, which will be automatically created upon deletion of the old event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the event log has been purged, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indication that the ground controllers will have that the event log was purged will be the new event log. The event log may be purged at any time, even if it has not been downlinked to the NPS ground station. The event log will be purged and a new log started automatically upon the completion of a successful downlink.

## TELEMETRY CONTROL

Operation: Purge the telemetry data stored in the flash memory.

Command: **purge\_flash\_tele**m

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>purge_flash_tele</b> m	Password (4 Bytes)
---	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **purge\_flash\_tele**m commands PANSAT to purge all of the telemetry data stored in the flash memory. Flash telemetry may be purged at any time, regardless of whether or not the telemetry has been downlinked.

Super User: Yes



Expected Response: **purge\_flash\_telem** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the flash memory has been purged and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the flash memory was purged, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communications session. At this point, the only indications that the ground controller will have that the flash memory has been purged will be in the event log.

## TELEMETRY CONTROL

Operation: Purge the telemetry data stored in static RAM.

Command: **purge\_sram\_telem**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>purge_sram_telem</b>	Password (4 Bytes)
---	-----------------------

AX.25 Information field

Required Parameters: None

Optional Parameters: None

Function: **purge\_sram\_telem** commands PANSAT to purge all of the telemetry data stored in the static RAM. SRAM telemetry may be purged at any time regardless of whether or not the telemetry has been downlinked.

Super User: Yes

Expected Response: **purge\_sram\_telem** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the stored telemetry has been purged and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the telemetry has been purged, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indication that the ground controllers will have that the stored telemetry has been purged will be in the event log.

## FILE SYSTEM CONTROL

Operation: Upload a file to PANSAT

Command: **put\_file** <Drive:\Path\Filename>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>put_file</b>	DOS File Path & Filename <Filename>	File Length (4 Bytes)	File Data (XXX Bytes)
-------------------------------------	--	--------------------------	--------------------------

AX. 25 Information Field

Required Parameters:     **<Filename>** The DOS path & filename to send to PANSAT. This is the path & filename as it exists on the ground station file system, **NOT** how it will appear on the PANSAT file system. This format allows ground controllers to retrieve and send files from any of the various sub-directories of the ground station hard drive thus keeping the ground station better organized.

Optional Parameters:     None

Function:                     The command **put\_<Filename>** is use to send or upload a specified file to PANSAT.

Super User:                  No

Expected Response:        PANSAT will not directly confirm or verify the command **put\_<Filename>** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the file directory and the event log for the appropriate indications.

## OS CONTROL

Operation: Read real-time clock

Command: **read\_clock**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_clock</b>
---------------------------------------

AX.25 Information Field

Required parameters: None

Optional Parameters: None

Function: **read\_clock** is used to read the clock on board PANSAT.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **read\_clock**. Receipt of the time information (in Universal Time Code format: yy/mm/dd/hh/mm/ss) should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **read\_clock** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_clock</b>	UTC (4 Bytes) <UTC>
---------------------------------------	------------------------

AX.25 Information Field

## EVENT LOG

Operation: Read the current event log.

Command: **read\_event\_log**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_event_log</b>
---

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **read\_event\_log** commands PANSAT to transmit the contents of the event log.

Super User: No



Expected Response: PANSAT will not directly confirm or verify the command **read\_event\_log**. Receipt of the event log should serve as the indication that the command was executed properly. PANSAT will first transmit the contents of the log and then make an entry that the command was executed. Hence, the transmitted event log will not show the most recent **read\_event\_log** command. If there is doubt as to whether or not the command was executed, ground controllers should examine subsequent logs. The protocol syntax for the response to the command **read\_event\_log** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_event_log</b>	Event Log (XYZ Bytes)
---	--------------------------

AX.25 Information Field

## TELEMETRY CONTROL

Operation: Read the telemetry data stored in the flash memory.

Command: **read\_flash\_telem**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_flash_telem</b>
---

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **read\_flash\_telem** commands PANSAT to transmit the telemetry data stored in the flash memory.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **read\_flash\_telem**. Receipt of the flash data should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **read\_flash\_telem** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) read_flash_telem	Length (4 Bytes) <Count>	Flash Telemetry Data ... (XXX Bytes)
--------------------------------------	-----------------------------	---

AX.25 Information Field

## MASS STORAGE UNIT

Operation: Read data from a specified flash memory location.

Command: **read\_flash\_mem** <Address>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_flash_mem</b>	Address (3 Bytes) <Address>
---	--------------------------------

AX.25 Information Field

Required Parameters: <Address> Specifies the flash memory location to begin reading data. n x 16 bytes of data will be read, where n defaults to 1 and may be changed to any value between 1 - 15 by changing the operating system parameters. A maximum of 15 x 16 bytes, or 240 bytes of information may be read with this command.

Optional Parameters: None

Function: The **read\_flash\_mem** command is used to read a specific flash memory location or a specific range of flash memory locations which may be beneficial to ground controllers performing troubleshooting.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **read\_flash\_mem**. Receipt of the flash memory contents should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log for the appropriate indications. The protocol syntax for the response to the command **read\_flash\_mem** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_flash_mem</b>	Address (3 Bytes) <Address>	Flash Memory Contents ... (240 Bytes Max.)
---	--------------------------------	---

AX.25 Information Field

## OS CONTROL

Operation: Read the Operating System parameters

Command: **read\_os\_param**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_os_param</b>
--

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: The **read\_os\_param** command is used to read the **current** parameters of the operating system. Operating system parameters (which are still to be determined) include **the** version of the current operating system, time and date, **and task** list.

Super User: No

Expected response: PANSAT will not directly confirm or verify the command **read\_os\_param**. Receipt of the parameters should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **read\_os\_param** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_os_param</b>	OS Parameter list (2 Bytes)
--	--------------------------------

AX.25 Information Field

## TELEMETRY CONTROL

Operation: Read the most recent telemetry

Command: **read\_recent\_telem**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_recent_telem</b>
--

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **read\_recent\_telem** commands PANSAT to transmit the current, most recent telemetry data.

Super User: No



Expected Response: PANSAT will not directly confirm or verify the command **read\_recent\_telem**. Receipt of the most recent telemetry data should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **read\_recent\_telem** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_recent_telem</b>	Length (2 Bytes)	Telemetry Data ... (X Bytes)
--	---------------------	---------------------------------

AX.25 Information Field

## MASS STORAGE UNIT

Operation: Read data from a specified static RAM location.

Command: **read\_sram\_mem** <Address>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_sram_mem</b>	Address (3 Bytes) <Address>
--	--------------------------------

AX.25 Information Field

Required Parameters: <Address> Specifies the SRAM address to begin reading data. n x 16 bytes of data will be read, where n defaults to 1 and may be changed to any value between 1 - 15 by changing the operating system parameters. A maximum of 15 x 16 bytes, or 240 bytes of information may be read with this command.

Optional Parameters: None

Function: The **read\_sram\_mem** command is used to read a specific SRAM location or a specific range of SRAM locations which may be beneficial to ground controllers performing troubleshooting.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **read\_sram\_mem**. Receipt of the SRAM contents should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log for the appropriate indications. The protocol syntax for the response to the command **read\_sram\_mem** is as follows:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_sram_mem</b>	Address (3 Bytes) <Address>	SRAM Memory Contents ... (240 Bytes Max.)
--	--------------------------------	--

AX.25 Information Field

## TELEMETRY CONTROL

Operation: Read the telemetry data stored in the static RAM.

Command: **read\_sram\_telem**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>read_sram_telem</b>
--

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **read\_sram\_telem** commands PANSAT to transmit the telemetry data stored in the static RAM.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **read\_sram\_telem**. Receipt of the stored telemetry data should serve as the indication that the command was executed properly. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log. The protocol syntax for the response to the command **read\_sram\_telem** is as follows:

Flag	Address	Control	PID	Info Field	CRC	Flag
01111110	112/560 Bits	8 Bits	8 Bits	256 Bytes	16 Bits	01111110

AX.25 Information Frame

Command (1 Byte)	Length (4 Bytes)	Telemetry Data File Contents...
<b>read_sram_telem</b>	<Length>	(X Byte)

AX.25 Information Field

## ELECTRICAL POWER SUBSYSTEM

Operation: Reset watchdog timer

Command: **reset\_watchdog**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information frame

Command (1 Byte) <b>reset_watchdog</b>
---

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **reset\_watchdog** is used to reset the watchdog timer on board PANSAT. Every 8 minutes, the selected DCS signals the watchdog timer that it is still functioning. If the selected DCS fails to signal the watchdog timer in the allotted time, the watchdog timer assumes that there is a problem with that particular DCS and switches to the alternate DCS. By issuing the **reset\_watchdog** command, ground controllers are assured

that they will be working with the same DCS for at least 8 minutes and thus simplifying any potential troubleshooting scenarios.

Super User: No

Expected Response: PANSAT will not directly confirm or verify the command **reset\_watchdog**. If there is doubt as to whether or not the command was executed, ground controller should examine the event log.

## COMMUNICATIONS UNIT

Operation:                      Select Transceiver parameters

Command:                      **sel\_trans\_param** <Control Byte>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>sel_trans_param</b>	Password (4 Bytes)	Control Byte (1 Byte) <Control Byte>
--	-----------------------	---

AX.25 Information Field

Required Parameters:

Not used	Not used	Mixer Bit !#	Mixer Bit #2	LNA Bit #1	LNA Bit #2	HPA Bit #1	HPA Bit #2
-------------	-------------	-----------------	-----------------	---------------	---------------	---------------	---------------

Control Byte

Bit combination 00 is not valid

Bit combination 11 is not valid

Bit combination 01 selects device #1

Bit combination 10 selects device #2



Optional Parameters:     None

Function:                    **sel\_trans\_param** is used to set the transceiver parameters. Each transceiver parameter (mixer, LNA, HPA) is controlled using two bits of the control byte. The two least significant bit positions are used to select the HPA. The next most significant bits (positions 3 & 4) are used to select the LNA and the next two significant bits (positions 5 & 6) are used to select the mixer. The two most significant bits (position 7 & 8) are not used in this command, are not read by the microprocessor, and set permanently to 0 for completeness. The example below shows mixer 1#, LNA 1#, and HPA 1# selected and is also the default control byte pattern.

Example:

Not used	Not used	0	1	0	1	0	1
-------------	-------------	---	---	---	---	---	---

Super User:                Yes

Expected Response:       **sel\_trans\_param** is considered a mission critical command and in addition to requiring a valid password, PANSAT will

query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the state of the transceiver has been changed and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the state of the transceiver has changed, PANSAT will not attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indications that the ground controllers will have that the state of the transceiver has changed will be through the event log.

## COMMUNICATIONS UNIT

Operation: Select Transmit Mode

Command: **select\_xmit\_mode** <Mode>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>select_xmit_mode</b>	Password (4 Bytes)	Xmit Mode (1 Byte) <Mode>
---	-----------------------	------------------------------

AX.25 Information Field

Required Parameters:   <Mode>   BPSK = 00  
  Spread Spectrum = FF

Optional Parameters:   None

Function:               The command **select\_xmit\_mode** is used to set the transmission mode of PANSAT to either BPSK or spread spectrum.

Super user:            Yes

Expected Response: PANSAT will not directly confirm or verify the command **select\_mode**. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log for the appropriate indications.

## OS CONTROL

Operation: Set the real-time clock on board PANSAT

Command: **set\_clock** <UTC>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 1028 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	--------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>set_clock</b>	Password (1 Byte)	UTC (4 Bytes) <UTC>
--------------------------------------	----------------------	------------------------

AX.25 Information Field

Required parameters: <UTC> In YY/MM/DD/HH/MM/SS format.

Optional Parameters: None

Function: **set\_clock** is used to update or reset the clock on board PANSAT. Ground controllers should enter the current UTC time for the required parameter <UTC>. The ground station software will then calculate the required processing and transmission time and adjust the input UTC accordingly, thus resulting in a more accurate clock on board PANSAT.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **set\_clock**. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log.

## TELEMETRY CONTROL

Operation: Set the telemetry polling rates

Command: **set\_polling\_rates** <Table #>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>set_polling_rates</b>	Password (4 Bytes)	Table data (??? Bytes)
--	-----------------------	---------------------------

AX.25 Information Field

Required Parameters: <Table #> See the PANSAT Software Detailed Design Document for a description of the <Table #> parameter format.

Optional Parameters: None

Function: **set\_polling\_rates** instructs PANSAT how often to sample the various telemetry points. The command **set\_polling\_rates** requires the ground controllers to specify a table number <Table #> to be used with the command. A table is essentially a matrix that lists all of the telemetry points and how often they are to be polled or sampled. Many tables will be constructed,

numbered and available for selection by the ground controllers depending on the current operational scenario. It should be understood that in order to change the polling rate of one specific telemetry point, either an existing table number must be edited or a new table number constructed since the entire table is what is transmitted and not just one entry within a specific table.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **set\_polling\_rates**. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log.



## COMMUNICATIONS UNIT

Operation: Select Power Level

Command: **set\_power\_level** <Level>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>set_power_level</b>	Password (4 Bytes)	Power level (1 Byte) <Level>
--	-----------------------	---------------------------------

AX.25 Information Field

Required Parameters: <Level> One of four discrete values, that are still to be determined, may be selected.

Optional Parameters: None

Function: The command **set\_power\_level** is used to set the transmitter power level on board PANSAT.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **set\_power\_level**. if there is doubt as to whether or not the command was executed properly, ground controllers should examine the event log for the appropriate indications.

## ELECTRICAL POWER SUBSYSTEM

Operation: Set power switches

Command: **set\_pwr\_switch** <Control Byte>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>set_pwr_switch</b>	Password (4 Bytes)	Control Byte (1 Byte) <Control Byte>
---	-----------------------	---

AX.25 Information Field

Required Parameters: At least one optional parameter

Optional Parameters:

Not Used	Not Used	Not Used	MSB	TMUX B	MSA	TMUX A	RF
-------------	-------------	-------------	-----	-----------	-----	-----------	----

Control Byte

**<rf = 1/0>** Toggle on/off RF system  
**<msa = 1/0>** Toggle on/off mass storage A  
**<msb = 1/0>** Toggle on/off mass storage B  
**<amuxa = 1/0>** Toggle on/off analog mux A  
**<amuxb = 1/0>** Toggle on/off analog mux B

Function: **set\_pwr\_switch** is used to toggle on/off the power switches to the five peripheral devices. Each bit position within the control byte corresponds to a specific peripheral device and by toggling these bits either on or off, ground controllers are able to control power to the peripheral devices. Toggling power to multiple peripheral devices may be achieved with a single **set\_pwr\_switch** command.

Super User: Yes

Expected Response: **set\_pwr\_switch** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that the state of a power switch has been changed and an entry will be made in the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that the state of a power switch has changed , PANSAT

will **not** attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indications that the ground controllers will have that the state of a power switch has changed will be in the event log.

## TELEMETRY CONTROL

Operation: Set the telemetry storage rates

Command: **set\_storage\_rates** <Table #>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>set_storage_rates</b>	Password (4 Bytes)	Table Data (??? Bytes)
--	-----------------------	---------------------------

AX.25 Information Field

Required Parameters: <Table #> See the PANSAT Software Detailed Design Document for a description of the <Table #> parameter format.

Optional Parameters: None

Function: **set\_storage\_rates** instructs PANSAT how often to store the various telemetry points. The command **set\_storage\_rates** requires the ground controllers to specify a table number <Table #> to be used with the command. A table is essentially a matrix that lists all of the telemetry points and how often

samples should be stored. Many tables will be constructed, numbered, and available for selection by the ground controllers depending on the current operational scenario. It should be understood that in order to change the storage rate of one specific telemetry point either an existing table number must be edited or a new table number constructed.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **set\_storage\_rates**. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log.

## TASK CONTROL

Operation: Start a software tasks on board PANSAT.

Command: **start\_task** <Task Name>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>start_task</b>	Password (4 Bytes)	Task Name <Task Name>
---------------------------------------	-----------------------	--------------------------

AX.25 Information Field

Required Parameters:     **<Task Name>** The name of the software task to be started and may be up to eight characters in length. All software tasks will have the same three-character extension.

Optional Parameters:     None

Function:                 **start\_task** is used to start a software task in the PANSAT task queue. Software tasks are executable programs that are written and compiled on the ground and instruct PANSAT how to perform under various circumstances.



Super User: Yes

Expected Response: **start\_task** <Task Name> is considered a mission critical command and in addition to requiring a valid pass word, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that a software task has been started and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that a software task was started, PANSAT will **not** attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, if there is doubt whether or not a task was started, ground controllers should execute a **list\_tasks** command to verify the contents of the task queue and the status of the task that was to have been started. Ground controllers may also examine the event log in order to confirm that the command was executed.

## TASK CONTROL

Operation: Stop a software tasks on board PANSAT.

Command: **stop\_task** <Task Name>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>stop_task</b>	Password (4 Bytes)	Task Name <Task Name>
--------------------------------------	-----------------------	--------------------------

AX.25 Information field

Required Parameters: **<Task Name>** The name of the software task to be stopped and may be up to eight characters in length. All software tasks will have the same three-character extension.

Optional Parameters: None

Function: **stop\_task** is used to stop a software task in the PANSAT task queue. Software tasks are executable programs that are written and compiled on the ground and instruct PANSAT how to perform under various circumstances.

Super User: Yes

Expected Response: **stop task <Task Name>** is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that a software task has been stopped and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that a software task was stopped, PANSAT will not attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indications that a software task has been stopped will be in the event log, or by ground controllers issuing a **list\_tasks** command to verify the contents of the task queue and the status of the task that was to have been stopped.

## OS CONTROL

Operation: Stop watchdog timer updates.

Command: **stop\_wdog**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>stop_wdog</b>	Password (4 Bytes)
--------------------------------------	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: The **stop\_wdog** command instructs the current system controller to stop sending updates to the watchdog timer.

Super User: Yes

Expected response: PANSAT will not directly confirm or verify the command **stop\_wdog**. If successful however, after **4/8** minutes, the watchdog timer will time out, thus resulting in the alternate system controller being booted up and brought on line. If there is doubt as to whether or not the command was executed, ground controllers should examine the event log.

## USER CONTROL

Operation: Allow new users.

Command: **unlock\_users**

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>unlock_users</b>	Password (4 Bytes)
---	-----------------------

AX.25 Information Field

Required Parameters: None

Optional Parameters: None

Function: **unlock\_users** is used subsequent to the **lockout\_users** command to restore normal user access to PANSAT. **unlock\_users** may be time tagged and used in conjunction with the **lockout\_user** command in order to restrict the number of users during eclipse and hence reduce power consumption, or to prepare PANSAT for an impending communications session with the NPS ground station.

Super User: Yes

Expected Response: PANSAT will not directly confirm or verify the command **unlock\_users** and unless there are indications to the contrary, it should be assumed that the command was executed properly. If there is doubt whether or not the command was executed, the ground controllers should analyze the event log for the appropriate indications.

## OS CONTROL

Operation: Update the current operating system parameters.

Command: **update\_os\_param** <Parameter List>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>update_os_param</b>	Password (4 Bytes)	Parameter List... (2 Bytes)
--	-----------------------	--------------------------------

AX.25 Information Field

Required Parameters: <Parameter List>

Optional Parameters: None

Function: The **update\_os\_param** command is used to update the operating system parameters. It is useful when the entire operating system need not be updated, but only revised, or changed slightly. A complete list of operating system parameters is still under development.

Super User: Yes



Expected response: PANSAT will not directly confirm or verify the command **update\_os\_param**. If there is doubt as to whether or not the command was executed, ground controllers may execute a **read\_os\_param** command and view the result or, examine the event log.

## MASS STORAGE UNIT

Operation: Write data to a specified flash memory location

Command: **write\_flash\_mem** <Address> <Data>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>write_flash_mem</b>	Password (4 Bytes)	Address (3 Bytes) <Address>	Data (1 Byte) <Data>
--	-----------------------	--------------------------------	-------------------------

Ax.25 Information Field

Required Parameters:      <Address> Specifies the flash storage address to begin writing the data. Valid range is from 0 - 0X7FFFF.

                                 <Data> The data to be written to flash storage. Valid range is from 0 - 0XFF.

Optional Parameters:      None

Function:	The <b>write_flash_mem</b> command is used to write data to a specific flash storage location. This command may be beneficial to ground controllers performing troubleshooting or attempting to repair a file that was somehow corrupted.
Super User:	Yes
Expected Response:	<b>write_flash_mem</b> is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that a flash storage location has been overwritten and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that a specific flash storage location has been overwritten, PANSAT will <b>not</b> attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indication that the ground controllers will have that a flash storage location has been overwritten will be in the event log.

## MASS STORAGE UNIT

Operation: Write data to a specified static RAM location

Command: **write\_sram\_mem** <Address> <Data>

Protocol Syntax:

Flag 01111110	Address 112/560 Bits	Control 8 Bits	PID 8 Bits	Info Field 256 Bytes	CRC 16 Bits	Flag 01111110
------------------	-------------------------	-------------------	---------------	-------------------------	----------------	------------------

AX.25 Information Frame

Command (1 Byte) <b>write_sram_mem</b>	Password (4 Bytes)	Address (3 Bytes) <Address>	Data (1 Byte) <Data>
---	-----------------------	--------------------------------	-------------------------

AX.25 Information Field

Required Parameters:      <Address> Specifies the sram storage address to begin writing the data. Valid range is from 0 - 0X3FFFF.

                                 <Data> The data to be written to sram storage. Valid range is from 0 - 0XFF.

Optional Parameters:      None

Function:	The <b>write_sram_mem</b> command is used to write data to a specific sram storage location. This command may be beneficial to ground controllers performing troubleshooting or attempting to repair a file that was somehow corrupted.
Super User:	Yes
Expected Response:	<b>write_sram_mem</b> is considered a mission critical command and in addition to requiring a valid password, PANSAT will query the NPS ground station prior to execution in order to verify its intentions. The NPS ground station will be immediately notified that a sram memory location has been overwritten and an entry will be made to the event log. If PANSAT moves over the horizon without being able to notify the NPS ground station that a specific sram storage location has been overwritten, PANSAT will <b>not</b> attempt to inform the NPS ground station of this event on subsequent communications sessions. At this point, the only indication that the ground controllers will have that a sram storage location has been overwritten will be in the event log.

## **APPENDIX B. A COMPLETE LISTING OF I/O PORTS AND THEIR DEFINITIONS**

<b>Port Address</b>	<b>Device</b>
0x000 - 0x07F	Serial Communications Controller (SCC)
0x080 - 0x0FF	Analog-to-Digital Converter (A/D)
0x100 - 0x17F	Programmable Peripheral Interface (PPI)
0x180 - 0x1FF	Paramax Control Register
0x200 - 0x27F	Paramax Register Interface

The microprocessor communicates with the other devices using the allocated port addresses. These address lines provide for the basic control and data I/O of these devices.



**APPENDIX C. A COMPLETE LISTING OF ADDRESSES AND  
SUBADDRESSES FOR PERIPHERAL DEVICES.**

<b>System Name</b>	<b>System Address</b>	<b>System Address</b>
RF System	0000, 0001	0, 1
Electrical Power System	1000, 1001	8, 9
System Control A	0010, 0011	2, 3
System Control B	1010, 1011	A, B
Analog MUX A	0100, 0101	4, 5
Analog MUX B	1100, 1101	C, D
Mass Storage A	0110, 0111	6, 7
Mass Storage B	1110, 1111	E, F





## LIST OF REFERENCES

1. Horning, J. A., *Navy Education Through Amateur Radio Satellite Development*, AIAA 93-4211, AIAA Space Programs and Technologies Conference and Exhibit, September, 21-23, 1993, Huntsville, AL.
2. Larson, Wiley, J. and Wertz, James, R., eds., *Space Mission Analysis and Design*, 2nd ed., p.380, Microcosom, Inc., 1992.
3. Lawrence, Greg, W., *Preliminary PANSAT Ground Station Software Design and Use of an Expert System to Analyze Telemetry*, Master's Thesis, Naval Postgraduate School, March, 1994.
4. Calvert, Thomas, C., *Computer Interface Development For The Petite Amateur Navy Satellite (PANSAT) Simulator*, Master's Thesis, Naval Postgraduate School, December, 1994.
5. Horning, J. A., *PANSAT Software Detailed Design Document (Draft)*, SSAG-D-PA:DC:001, Naval Postgraduate School, August 23rd, 1995.
6. Bible, Steven, R., *The World Wide Webb Amateur Satellite Ground Station*, The AMSAT Journal, Vol. 18, No. 3, Silver Springs, MD., May/June, 1995.
7. Stallings, W., *Data and Computer Communications*, 4th ed., p. 145, Macmillan Publishing Company, 1994.
8. Peterson, W., and Brown, D., *Cyclic Codes for Error Detection*, Proceedings of the IRE, January 1961.



## INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5101	2
3. Chairman, (Code SP) Space Systems Academic Group Naval Postgraduate School Monterey, CA 93943-5000	1
4. I. Michael Ross, (Code AA) Naval Postgraduate School Monterey, CA 93943-5000	3
5. Dan Sakoda, (Code SP) Space Systems Academic Group Naval Postgraduate School Monterey, CA 93943-5000	1
6. Jim Horning (Code SP) Space Systems Academic Group Naval Postgraduate School Monterey, CA 93943-5000	1
7. Troy M. Nichols 1407 Meadow Dr. Walled Lake, MI 48390	2